2025 Fall CSED451 Assignment1 Report

Team Name: openGameLab

Team Member #1: 안재영

컴퓨터공학과/20220019/enter21

Team Member #2: 정윤혁

컴퓨터공학과/20240385/jyhyeok

1. Technical Details

개발 환경: Visual Studio 2022 버전 17.14.9 (July 2025)

cpp 확장자 파일을 powershell 스크립트 파일로 만든 후 powershell에서 빌드하여 구현 결과를 확인할 수 있다. 또한, OpenGL을 사용하는 코딩에 익숙해야 한다.

2. Implementation Details

2-1. Outline

drawPlayer_, drawSquare, drawCircle, drawBoss 이 네 가지 drawing 기능을 가진 함수들을, drawPlayer, drawEnemy, drawBullets, drawText에서 사용한다. 그리기는 모두 display 함수에서 담당한다. 이 display 함수에서 게임에 등장하는 모든 오브젝트를 그린다. Double buffer를 이용하여 깜빡임 등의 현상이 일어나지 않도록 한다.

wasd키를 키다운하여 움직일 수 있다. 키다운을 이용했기에 두 개 이상의 키를 동시에 키다운하여 4방향뿐만 아니라 대각선 등 모든 방향으로 이동할 수 있다. 스페이스바를 키다운하면 Bullet을 직선 방향으로 쏘아 공격한다. 플레이어 이동 및 공격을 위한 키 입력은 processInput, handleKeyDown, handleKeyUp 함수들로 관리한다. 추가적으로 processInput에서 플레이어가 윈도우 창 외부로 나갈 수 없도록 한다.

Bullet과의 충돌은 rectCollision 함수로 감지하고, 충돌 시 내부 처리는 handleCollisions 함수에서 맡는다. Bullet들 각각에 isFromPlayer라는 bool타입 변수를 부여하여, 플레이어의 Bullet이 적과 충돌했을 경우, 적의 Bullet이 플레이어와 충돌했을 경우를 구분한다. 플레이어의 경우 충돌 시 life가 감소하고 무적 시간 동안 잠깐 화면에서 사라진 후 리스폰하고, 적의 경우 HP가 감소한다. 플레이어의 life나 적의 HP가 0이 되면 게임이 종료되고 적절한 텍스트를 출력한다.

Bullet의 처리는 spawnEnemyBullet, updateBullets으로 처리하며, 두 함수는 timer 함수에서 사용되며 추가로 timer 함수는 리스폰 기능도 관리한다. spawnEnemyBullet은 딱 하나만 존재하는 적으로부터 Bullet을 생성하는 함수이다. updateBullets는 vector로 저장한 Bullet들을 적절히 업데이트하여 Bullet의 위치를 이동시키고 창 밖으로 나간 Bullet들을 없앤다. timer에서는 위의 두 함수를 이용해 Bullet을 핸들링하며 플레이어 리스폰 기능도 존재한다.

main에서는 플레이어 life, 시작 포지션 등 변수들을 초기화하고, GLUT과 GLEW를 initialize한다. GLUT을 이용하여 키보드 입력, 윈도우 창 생성, 게임 루프 기능을 사용한다. OpenGL 기본 설정들로 창 배경색, 행렬 초기화, 좌표계 설정을 진행한다. initializeVA라는 함수를 이용하여 vertex 배열들을 initialize하는데, 이는 다음에 다루겠다.

2-2. Additional Requirements

handleKeyDown에서 키보드 R키를 누르면 재시작하는 기능을 추가로 구현하였다.

플레어이와 적을 단순하게 삼각형이나 사각형으로 만들지 않고, 점을 여러 개 이용하여 좀 더 복잡한 모양으로 만들었다. 플레이어의 경우 삼각형과 사각형을 적절히 조합하여 전투기와 비슷한 모양을, 적의 경우 원과 별을 조합하여 항공모함과 비슷한 모양으로 만들었다. 또한 GL_QUADS나 GL_TRIANGLES로 간단하게 그릴 수 있는 기본 도형인 사각형, 삼각형과는 달리, 원과 별의 경우 OpenGL에서 직접적으로 그릴 수 없기 때문에 GL_TRIANGLE_FAN과 삼각함수 연산을 이용하여 그렸다. 결과적으로 단순한 모양만을 렌더링하는 것이 아닌, 좀 더 복잡한 모양 또한 렌더링하였다.

렌더링 최적화를 위해 initializeVA라는 함수를 구현하였다. OpenGL에서 제공하는 라이브러리 함수들로도 이번 과제의 요구 사항을 만족시키는 데에는 큰 문제가 없지만, 이후 프로젝트가 더 커지게 된다면, CPU에서 점 정보를 GPU로 넘기고 GPU에서 그리는 OpenGL에서 제공하는 glVertex2f와 같은 함수의 방식은 깜빡임, 깨짐, 버퍼링 등 다양한 문제를 초래할 수 있다. 그러한 문제를 해결하기 위해, glVertex2f로 점의 좌표를 그때그때 받는 것이 아니라, initializeVA에서 미리 저장해둔 점들의 좌표를 그때그때 받아오는 방식을 채택했다. 이 함수에서는 플레이어, 사각형, 원, 별을 그리기 위해 필요한 좌표들을 미리 모두 계산해둔 후 배열에 저장해둔다. 이후 그리기 기능을 가진 함수들에서 적절히 점들의 좌표값들을 받아와 화면에 그리고, 점의 이동은 행렬 연산으로 처리해주면 CPU에서 GPU로 정보를 넘기는 과정이 보다 나아져 렌더링을 최적화시킬 수 있다.

Global camera animation으로 진동 효과를 display 함수에 추가로 구현하였다. 플레이어나 적이 Bullet에 의해 데미지를 받으면 shakeTimer 변수가 0에서 15로 변경된다. 이는

5프레임간 진동한다는 의미이다. 또한 rand를 사용하여 화면 전체를 랜덤하게 이동시켜, 진동 효과를 주었다. shakeTimer 변수는 --연산자로 점차 감소하게 구현하여, 일정 시간 동안만 화면이 진동하도록 하였다.

2-3. Design Rationale

Bullet과 Enemy를 struct 형식으로 정보를 저장했는데, 이는 Bullet과 Enemy에 필요한 다양한 변수들을 하나로 묶어 프로그램에서 사용하고자 하는 Bullet과 Enemy를 정의하기 위해서이다.

draw함수들을 다양하게 나눈 이유는 이후 확장성을 위해서이다. 기본 도형을 draw하는 함수를 오브젝트들을 draw하는 함수에서 사용하는 구조를 취했기에, 이후 만약 추가적으로 도형을 그려야 할 필요가 있거나 오브젝트를 추가할 때 유지보수가 쉽도록 하였다.

handleKeyDown, handleKeyUp으로 키 입력을 관리하는 이유는, 8방향 및 매끄러운 이동을 위해서이다. 만약 단순히 키를 눌렀을 때 그쪽 방향으로 위치를 이동시키는 방법을 채택한다면, 마치 텔레포트하는 듯한 느낌을 줄 수 있고 전방향 이동을 구현하기 까다로워져 구현의 측면에서도 플레이의 측면에서도 부정적 경험을 줄 가능성이 높다고 판단했기 때문이다.

2-4. Design Implementation

Bullet과 Enemy를 각각 struct로 선언한다. Bullet에는 좌표, 벡터, 스피드, 충돌 시 처리를 위한 isPlayerFrom bool 타입 변수가 존재한다. Enemy에는 좌표, 크기, HP, Bullet 발사 쿨타임인 shootCooldown, 게임 종료를 관리하기 위해 적이 살아있는지 죽었는지 판별하는 bool타입 변수 isAlive가 존재한다.

화면에 존재하는 Bullet들을 관리하기 위해 bullets라는 이름의 vector를 선언한다. Bullet이 생성되면 Bullets에 추가되고, 화면 밖으로 나가면 Bullets에서 삭제된다. Bullet을 관리할 때 Bullets를 파라미터로 for문을 돌려 Bullet들을 관리한다.

Enemy struct를 기반으로 적 오브젝트 enemy를 선언한다.

플레이어 공격, 리스폰, 카메라 진동 등을 위한 다양한 변수들을 선언한다.

점들의 정보를 저장한 vertex array들을 GLfloat로 선언한다. 이는 이후 initializeVA 함수에서 값들이 할당된다.

initializeVA에서는 vertex array들에 초기값을 할당한다. 삼각형과 사각형은 OpenGL의 함

수들로 그냥 그리면 되지만, 원과 별은 그럴 수 없다. 따라서, 이후 GL_TRIANGLE_FAN을 이용해 삼각형들의 집합으로 원과 별을 그리기 위해 점들의 위치를 삼각함수를 이용해 계산한다. 원의 경우 삼각형 36개로, 별의 경우 삼각형 10개로 그린다.

drawPlayer_, drawSquare, drawCircle, drawBoss는 공통적으로 glEnableClientState로 점들의 정보를 받아올 배열을 사용하겠다고 선언하고, glVertexPointer로 점 데이터들의 위치를 지정해준다. glPushMatrix로 변환 행렬을 저장한다. glScalef로 객체 크기 스케일링을 한다. glDrawArrays로 각 함수의 역할에 맞게 적절히 도형을 그린다. glPopMatrix로 변환 행렬을 초기화한다. glDisableClientState로 glEnableClientState 상태를 종료한다.

drawPlayer, drawEnemy, drawBullets, drawText에서는 위의 draw 함수들을 이용하는 함수이다. 공통적으로 glPushMatirx로 변환 행렬을 저장하고, glTranslatef로 파라미터만큼 점을 이동시킨다. glColor3f로 색상을 지정해준다. 함수의 의도에 맞는 적절한 draw함수를 사용한 후, glPopMatrix로 변환 행렬을 초기화해준다. drawEnemy에서는 적의 HP를 보여주기 위해 추가적으로 glBegin(GL_QUADS)와 glVertex2f로 적절한 점의 위치를 파라미터로 주어 HP바와 남은 HP를 표시를 표시한다. drawBullets에서는 vector Bullets를 순회하며 Bullet들을 그린다. 플레이어의 Bullet은 노랑색 직사각형 두 개, 적의 Bullet은 빨간색 원으로 그린다. drawText에서는 텍스트를 파라미터로 받은 후 char 단위로 순회하여, glutBitmapCharacter을 이용해 화면에 나타낸다.

위의 모든 draw함수들은 최종적으로 display함수에서 사용된다. 카메라 진동, 플레이어/적/Bullet/현재 플레이어 life와 적의 HP/게임 오버 텍스트/클리어 텍스트 그리기가 display 함수의 기능이며, double buffer를 이용하기 위해 이곳에서 glutSwapBuffers를 사용한다.

spawnEnemyBullet에서는 적에게서 Bullet이 생성될 때 시점에서 적과 플레이어의 좌표를 기반으로 방향 벡터를 계산하여 그쪽으로 Bullet을 발사하게 한다. updateBullets에서는 bullets vector를 순회하여 각 Bullet의 스피드와 방향 벡터를 기반으로 Bullet들의 위치를 갱신하고, 화면 밖으로 나가는 Bullet들을 삭제한다. 화면 밖으로 나가는 기준은, +-1.1f를 기준으로 했다. 1.0f 대신 1.1f로 한 이유는, 1.0f로 할 경우 Bullet의 크기가 0이 아니기 때문에 화면 모서리에서 갑작스럽게 없어지는 느낌을 주기 때문이다.

rectCollision에서는 사각형의 중심 좌표와 사이즈를 파라미터로 받아, 두 사각형이 x축과 y축 모두에서 겹치면 충돌한 것으로 판정한다. handleCollisions에서는 vector bullets를 순회를 돌면서 플레이어의 Bullet이 적과 충돌했는지, 적의 Bullet이 플레이어와 충돌했는지를 모두 판별한다. 충돌하지 않았을 경우 넘어가지만, 충돌한 Bullet의 경우 대상의 life나 HP를 감소시키고 카메라 진동 효과를 주기 위한 shakeTimer를 15로 설정해주며 해당 Bullet을 삭제한다. 또한 해당 충돌로 게임이 종료될 경우 그를 위한 bool타입 변수들의

값도 변경해준다.

keyState는 키 입력 상태를 관리한다. 이것은 이후 processInput, handleKeyDown, handleKeyUp에서 사용한다.

processInput은 게임 종료 상태거나 플레이어가 무적 상태라서 화면에 그려지지 않을 때는 입력을 받지 않기 위해 작동하지 않는다. 해당 bool타입 변수들을 if문에 파라미터로 넣어 해당 케이스에는 바로 return을 하는 방법을 채택했다. Wasd가 키다운 되어 있을 때 해당 방향으로 속도를 dx와 dy 변수를 이용해 넣어준다. 이를 연속해서 계산하여 플레이어의 위치를 업데이트한다. 또한, 플레이어가 화면 밖으로 나가지 않을 경우에만 이동이 가능하도록 하여, 플레이어가 화면 밖으로 나가는 상황을 방지했다. (플레이어 위치) + (플레이어 크기)가 기준 좌표계의 1.0f와 -1.0f를 넘지 않는 경우에만 움직이도록 하였다. 또한 플레이어가 스페이스바를 키다운 하는 중에 Bullet이 발사되도록 한다. 다만, 연발은 되지 않도록 쿨다운을 playerFireCoolDown이라는 이름의 bool타입 변수로 설정하여 쿨다운 중에는 그 변수를 --연산자로 계속 감소시키고, 발사가 가능할 때만 (playerFireCoolDown > 0 이 아닐 때) Bullet을 생성한다. 플레이어의 Bullet은 위쪽 수직 방향으로만 이동하고, 좌우 방향으로는 이동하지 않는다.

handleKeyDown은 키다운 중인 키를 keyState에서 true로 바꾸어 그 키가 현재 키다운 중임을 저장하고, 그 키가 키다운 중일 때의 처리를 하도록 한다. 또한, 키보드의 R키를 눌렀을 때 게임의 초기값들을 초기화하여 게임을 재시작할 수 있도록 한다.
handleKeyUp은 handleKeyDown과 반대로, 키다운을 해제한 키의 keyState를 false로 바꾸어 이제는 더 이상 그 키가 키다운 중이 아님을 저장하고, 그 키를 키다운 했을 때 처리를 하지 않도록 한다.

timer 함수는 게임 오버 상태가 아닐 때 키 입력을 받는다. 또한 플레이어와 적의 Bullet 생성을 관리한다. updateBullets와 handleCollision 함수도 실행하여 Bullet들의 상태와 충돌 시 처리를 담당한다. 추가적으로, 플레이어가 적의 Bullet과 충돌했을 때 플레이어를 리스폰시키는 기능도 담당한다. 리스폰 시 플레이어는 시작 지점으로 이동한다. 만약 남은 life가 0이 되었다면, 게임 오버시킨다.

main함수에서는 플레이어 위치, 적 위치, 플레이어 life, 각종 bool타입 변수들 등 게임에 필요한 변수들을 초기화시킨다. 또한 GLUT와 GLEW를 initialize하여 위의 함수들에서 OpenGL의 함수들뿐만 아니라 GLUT과 GLEW를 사용할 수 있도록 한다.
glutInitDisPlayMode에서 GLUT_DOUBLE으로 더블 버퍼를 사용함으로써 깜빡임을 방지한다. 또한 다른 파라미터인 GLUT_RGB로 컬러 모드를 사용한다. glutInitWindowSize에서 윈도우 창 크기를 800 x 600으로 설정한다. glutCreatWindow로 창 이름을 ASSN 1으로 설정한다. initializeVA 함수를 호출하여 점들의 위치를 초기화한다. glutDisPlayFunc 함수

로 화면을 그릴 때 display 함수가, glutKeyBoardFunc으로 키보드를 누를 때 handleKeyDown 함수가, glutKeyBoardUpFunc으로 키보드를 누른 것을 뗄 때 handleKeyUp 함수가, glutTimerFunc으로 일정 주기마다 timer함수가 호출되도록 설정한다. glClearColor로 화면 배경을 검정색으로 설정하고, glMatrixMode(GL_PROJECTION)으로 projection matrix 모드를 사용하고, glLoadIdentity로 행렬을 초기화하며, gluOrtho2D로 윈도우 창 화면을 2D 좌표계로 설정한다. 마지막으로 glutMainLoop로 루프를 실행하여 상술한 함수들이 반복해서 실행되도록 한다.

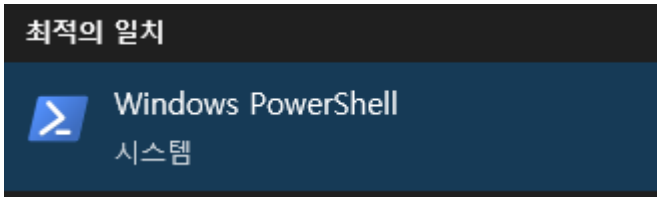
2-5. Additonal Background

OpenGL뿐만 아니라, GLUT을 initialize함으로써 사용할 수 있는 함수들을 미리 알고 있다면 코드를 이해하는 데 걸리는 시간이 더 짧아질 수 있다.

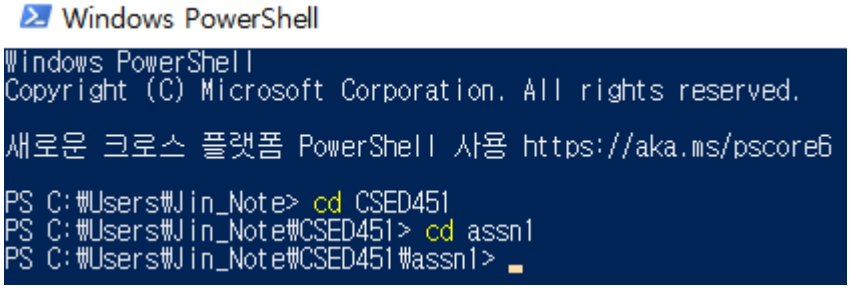
삼각함수 및 행렬에 대한 이해도가 있다면 점 위치 연산과 점 이동에 사용하는 변환 행렬들을 사용하는 코드들을 이해하기 쉬워질 수 있다.

3. End-User Guide

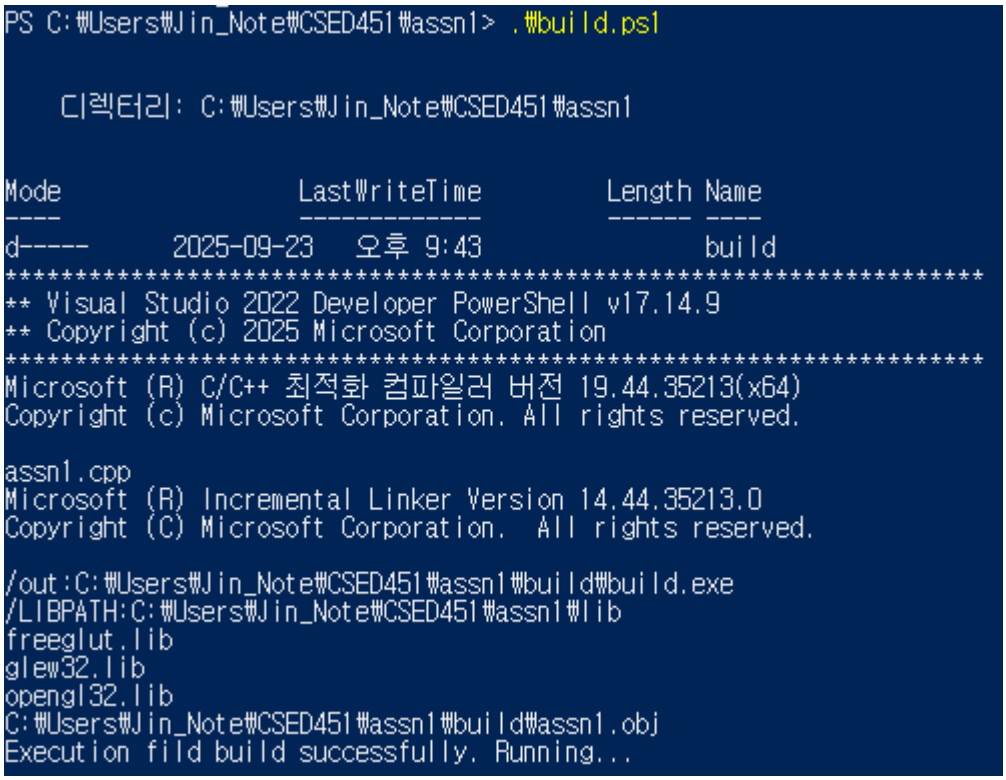
Windows powershell을 실행한다.



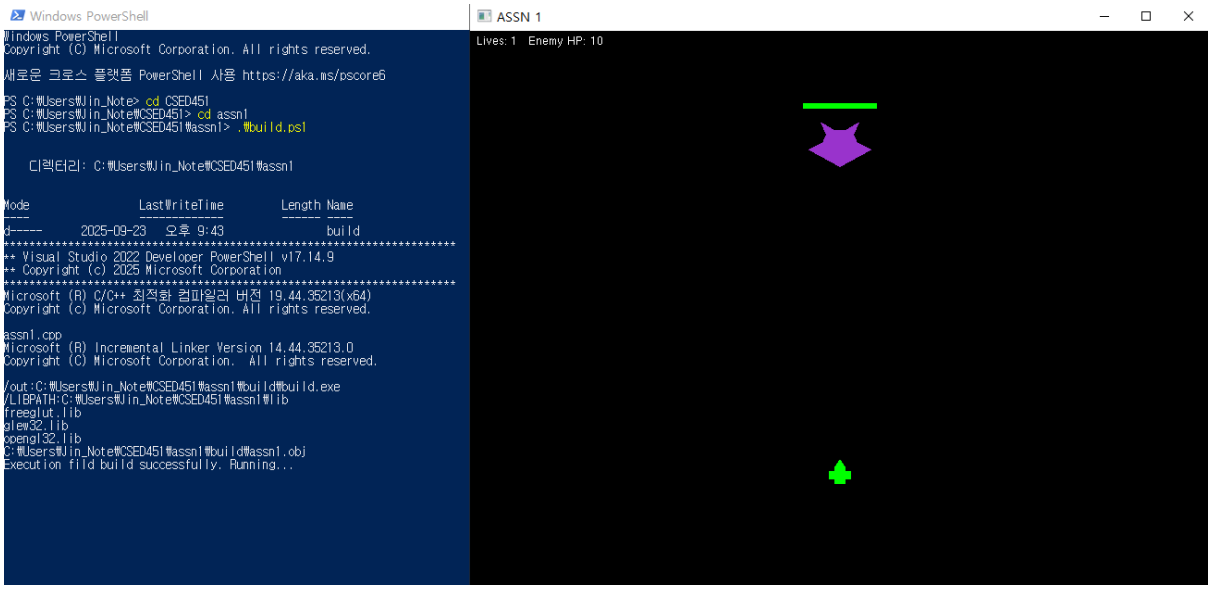
프로젝트가 저장된 파일 위치로 이동한다.



.₩build.ps1을 입력한다.



잘 실행되는 것을 확인할 수 있다.



프로그램은 위와 같은 형태의 윈도우 창으로 팝업되며, WASD 키를 사용해 움직임을 수행하고 스페이스바를 홀드하여 총알을 발사할 수 있다. 플레이어는 방향키로 적이 발사하는 총알을 회피하며 총알을 명중하여 모든 체력을 깎아야 한다. 플레이어의 목숨이 다 하였거나 게임 도중에도 재시작을 원할 경우, R키를 눌러 다시 시작할 수 있다.

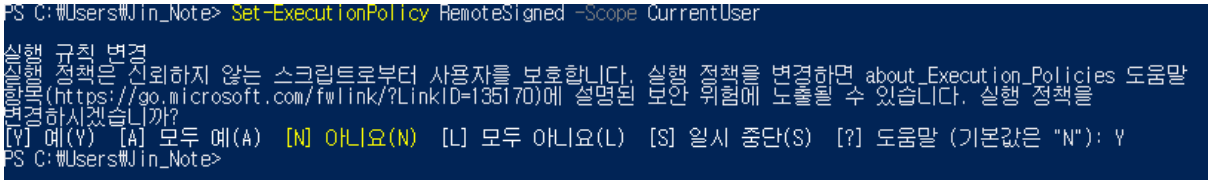
프로그램 종료 시 powershell 창에는 아래 메시지가 출력된다.

```
Execution exited.

PS C:₩Users₩Jin_Note₩CSED451₩assn1>
```

이후 다시 빌드하여 프로그램을 실행시키는 것 또한 가능하다.

만약 Powershell 스크립트 실행 정책이 제한되어 있는 등의 이유로 Powershell에서 스크립트를 실행할 수 없다면, Get-ExecutionPolicy 명령어를 입력하여 실행 정책이 제한되어 있는지 확인해야 한다. 만약 그렇다면, Set-ExecutionPolicy RemoteSigned -Scope CurrentUser 명령어를 입력하여 스크립트 실행 정책을 변경해야 한다. 정책 여부 변경을 묻는 메시지가 나타나면 Y를 입력하여 스크립트 실행 정책을 변경하면 실행이 잘 된다.



4. Discussions/Conclusions

4-1. Obstacles

- 다중 키 입력 처리 : 기본적으로 AI를 사용해 구현한 키 입력 처리 방식은 한 번에 하나의 키만 인식하는 구조였기 때문에, 동시에 여러 키를 눌러 이동 중 총알 발사나 대각선 이동을 구현하기 위해 키 입력 상태를 누름, 뗌에 따라 처리하는 로직을 추가해야 했다.

- 도형 렌더링 처리 : 가장 단순하게 draw 함수 호출 시에 그릴 좌표 및 각각의 점들을 하나씩 전달하는 방식을 기본으로, 원점을 기준으로 그리는 함수를 translate한 위치에서 호출하고 Vertex Array를 활용해 전체 점을 한번에 전달하는 방식으로 단계적으로 개선하는 과정을 거쳤다.

- OpenGL의 애니메이션 업데이트 방식 : timer와 display 등 함수들 안에서 게임씬을 구현하는 방법을 공부해야 했고, 부활 등 각종 이벤트에 적절한 프레임 수를 할당하는 데에 테스트가 필요했다.

4-2. Improvement Idea

- 모듈화 및 클래스 구조 : 현재는 대부분의 기능이 함수와 전역 변수 및 구조체로 정의되어 있으므로, 추후 구조체를 클래스화하고 객체지향적으로 함수들을 리팩토링하면 확장성 및 가독성이 향상될 것으로 보인다.

- 사운드 효과 추가 : 카메라 진동이 호출되는 피격 시에 추가로 사운드 효과를 추가하면 게임의 몰입감을 높일 수 있을 것이다.

- 적 패턴 심화 : 현재 적은 발사 시점에 플레이어와의 방향 벡터를 계산하여 하나의 총알을 발사하도록 구현되어 있는데, 한번에 여러개의 총알을 발사하거나 날아가는 동안에도 실시간으로 플레이어 방향을 연산하여 방향을 수정하는 유도탄 등을 추가하면 더 고도화된 게임성을 구현할 수 있을 것이다.

4-3. Lessons

이번 과제를 통해 다음과 같은 점들을 배울 수 있었다.

- OpenGL의 기본 작동 방식 이해 : OpenGL을 활용해 각종 도형을 렌더링하고 게임의 진행을 구현하기 위해 필요한 각종 내용들을 학습할 수 있었다.

- 입력 처리와 게임 루프의 이해 : 보다 구체적으로 키를 입력받고 타이머를 기반으로 게임 흐름을 루프시키는 구현하는 방식을 학습할 수 있었다.

5. References

Learn OpenGL - https://learnopengl.com/

6. AI-assisted Coding References

이번 과제의 약 65%가 AI(ChatGPT)의 도움을 받아 구현되었다. OpenGL을 활용한 게임 개발은 처음이었기 때문에, 기본적인 기능 구현에 있어 다음과 같은 항목에서 AI를 적극 활용하였다.

- 도형 렌더링 : OpenGL에서 삼각형, 사각형 등 기본 도형을 화면에 출력하는 방법을 학습할 때 AI의 코드 예시를 참고하였다. 다만 생성된 방법은 Vertex2f를 사용해 점들을 일일이 전달하는 방식이었기 때문에, 추후 Vertex Array를 사용하도록 개선하였다.

- 키보드 입력 처리 : 사용자의 키 입력을 감지하고 이에 따라 캐릭터가 움직이도록 하는 로직을 구성할 때, 이벤트 처리 방식과 관련된 코드를 AI의 설명을 기반으로 구현하였다. 초기의 방식은 한 번에 하나의 키 입력만을 받을 수 있었으므로, 추가 질문을 통해 키를 누르고 떼는 것을 분리하여 처리할 수 있음을 알고 다중 입력을 지원하도록 구현할 수 있었다.

- 애니메이션 구현 : 캐릭터가 지속적으로 움직이는 애니메이션을 구현하기 위해, 타이머와 프레임을 기반으로 하는 업데이트 방식에 대해 AI의 설명을 참고하였다. 이를 바탕으로 게임의 전체적인 흐름과 상태 업데이트 로직을 구성할 수 있었다.

위와 같은 기능들의 구현 방식을 AI를 통해 학습해 게임의 프로토 타입을 구현할 수 있었고, 이후 개별 오브젝트의 모델링 개선, 크기와 시간 파라미터 조절, 최적화 및 리팩토링 과정을 거쳐 프로젝트를 완성하였다.

7. individual contribution

이번 프로젝트는 두 팀원이 50:50 비율로 각자의 역할을 나누어 충실히 수행하였다. 각각 기본적인 게임의 구성 요소 및 카메라 애니메이션을 구현하는 역할, 오브젝트 모델링을 고도화하고 렌더링을 최적화하는 역할로 구현 파트를 나누고 각각에 맞는 보고서를 작성하여 프로젝트를 성공적으로 완수하였다.